Apache Ignite, взгляд изнутри

от распределения данных в кластере до операции записи на диск

Дмитрий Павлов





Dmitriy Pavlov



VP, Apache Ignite

Committer, PMC member



Apache Training (incubating)

Committer, PPMC member

SBER TECH

Customer Success Team lead

Chief Technology Expert



Ignite <summit>

Program committee member



14/18 years in Java development

dpavlov@apache.org



d-pavlov



Disclaimer

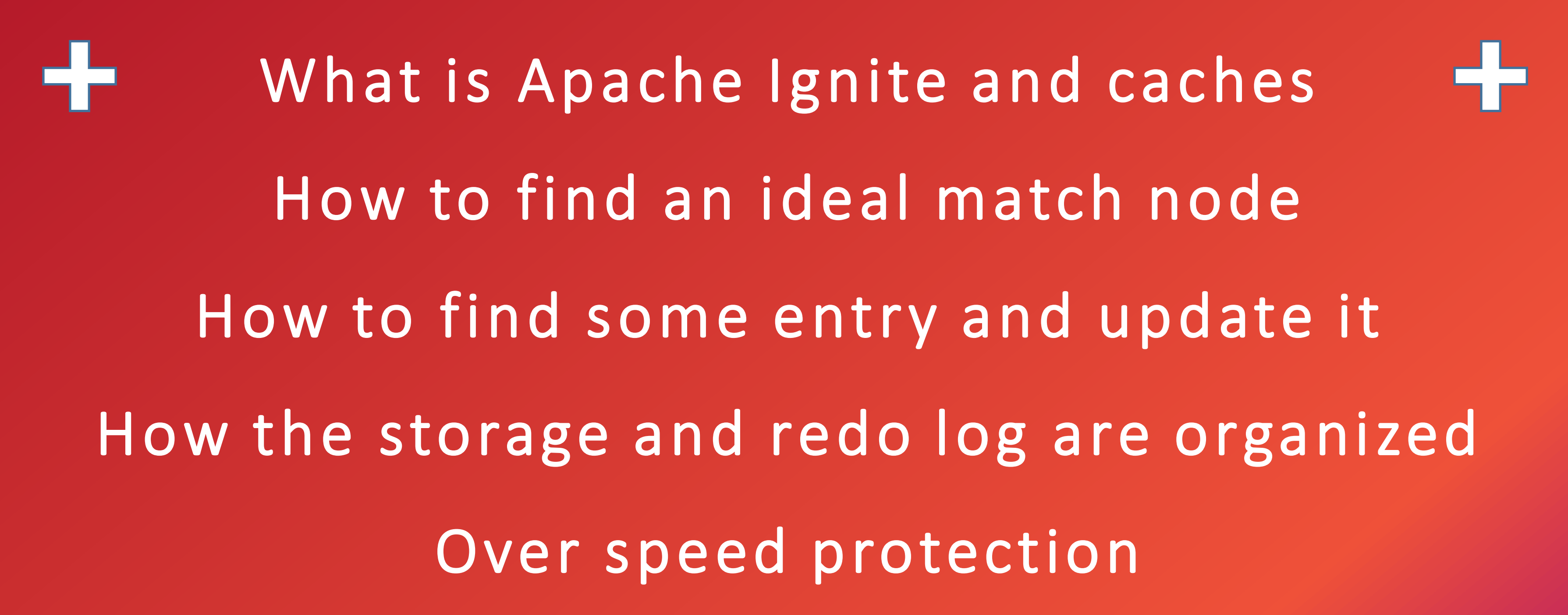
This talk represents my own personal view and opinion.

It does not necessarily reflect the official stance of

The Apache Software Foundation/SberTech/

any company/any other entity the author might be affiliated with at the moment of presenting or in the past.




AGENDA
What is Apache Ignite and caches
How to find an ideal match node
How to find some entry and update it
How the storage and redo log are organized
Over speed protection

Apache Ignite
Intro



is a Distributed Database
for High-Performance Computing
with In-Memory Speed

1 Ignite cluster – N Caches

Cache - key-value storage
- put(k,v)
- v=get(k)

In Memory Data Grid – yes
In Memory Database - yes
SQL support - yes
SQL database – not fully



IN-MEMORY DATA GRID

= Stores data in-memory

(data grid)

+ Compute - code goes to data

(compute grid)

DISTRIBUTED DATABASE

- Memory-centric database, since V2.1
- Scalable: Each node stores only it's own data part

Ignite can be used in combined mode (part in-memory, part - persisted)

External data source (DB, REST, other)

- Yes - Cache
- No - Ignite is Primary storage



Ignite Cluster

Compute &
Data Nodes

DURABLE MEMORY
ON-DISC

Ignite
Cluster

Client
Connectors

Public & Private
Clouds

- Servers – can store data
- Clients (thick) – put/get
- + Thin clients
- data distribution is handled by affinity function (Rendezvous Hashing)

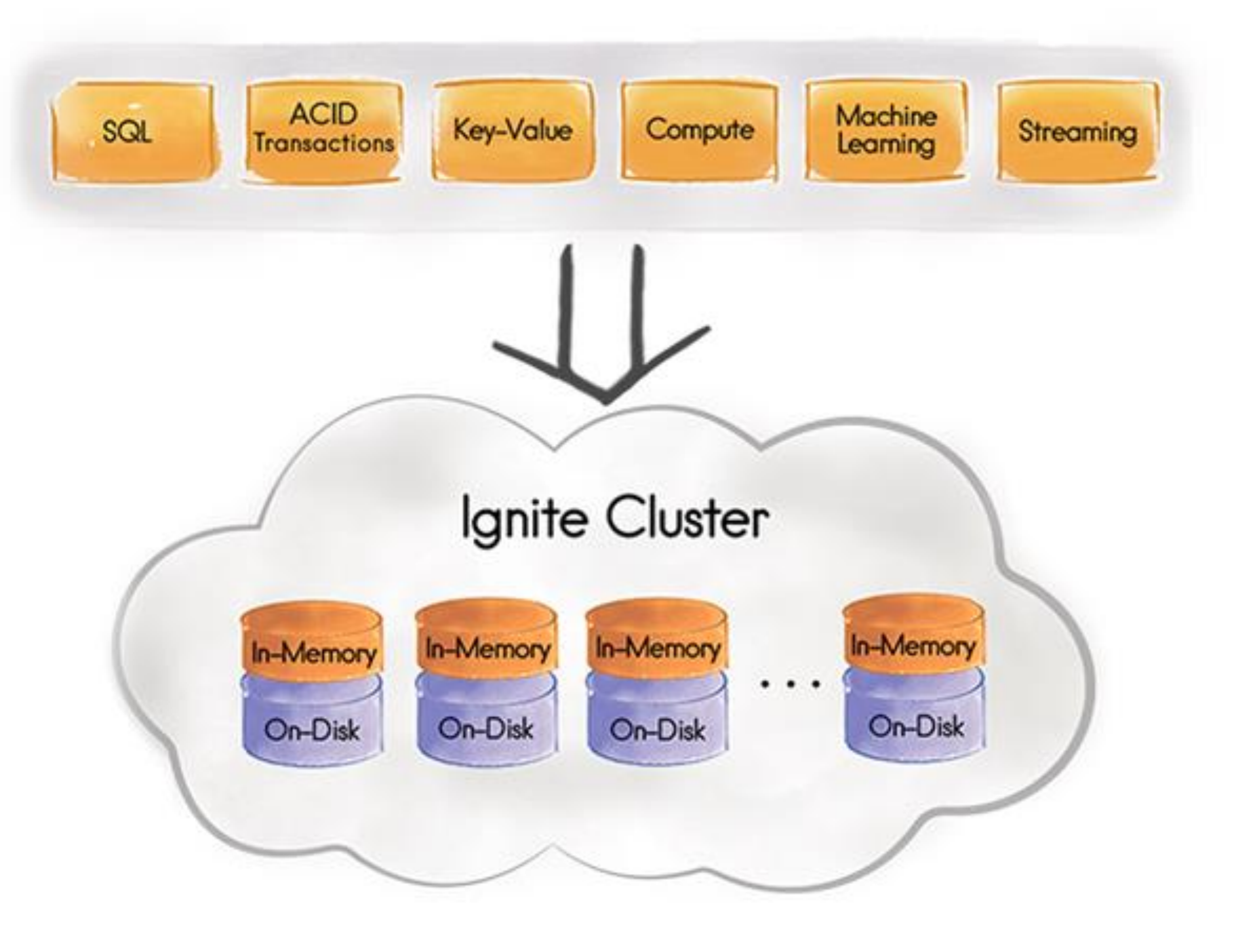


Apache Ignite Native Persistence

is a distributed ACID and SQL-compliant disk store

All data is on-disk,
part of the data is in-memory

Apache Ignite =
Speed/Scale



Each node has its own local storage

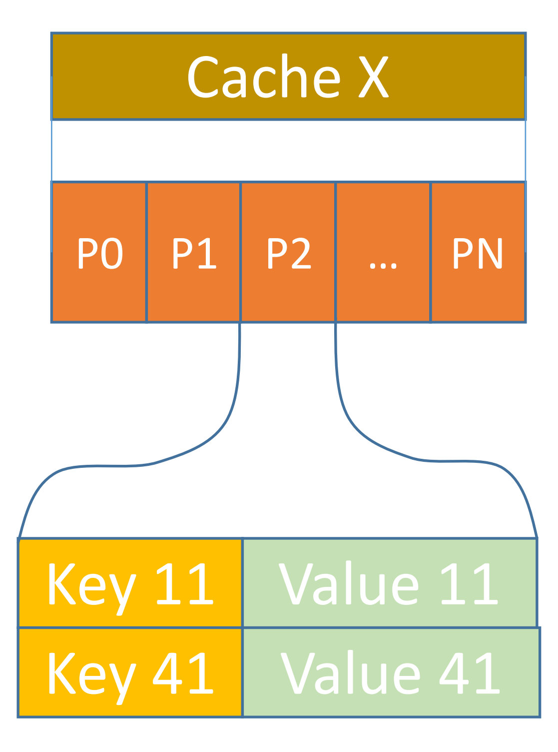


Apache Ignite
Cache

Ignite Cache

- Cache main feature: K->V
- Cache ~ Java.util.Map
- JSR-107, JCache API
- Cache ~ Table
- Cache usually stores 1 business entry

- Entry = K + V
- 1 Cache – N partitions
- K -> hash(K) ->partition -> node





Ignite Cache types

- Partitioned and Replicated
  - Replicated - Cache A
  - Partitioned - Cache B & C

- Replicated, use case:

rare write, often - read, e.g. dictionary

- Partitioned – most common

1024 default

- Backups 0,1...

Replicated

Partitioned

Cache A

Cache B

Cache C

Node 0

Node 1

C C B B B Cache A

C C C B B Cache A



AGENDA

What is Apache Ignite and caches

How to find an ideal match node

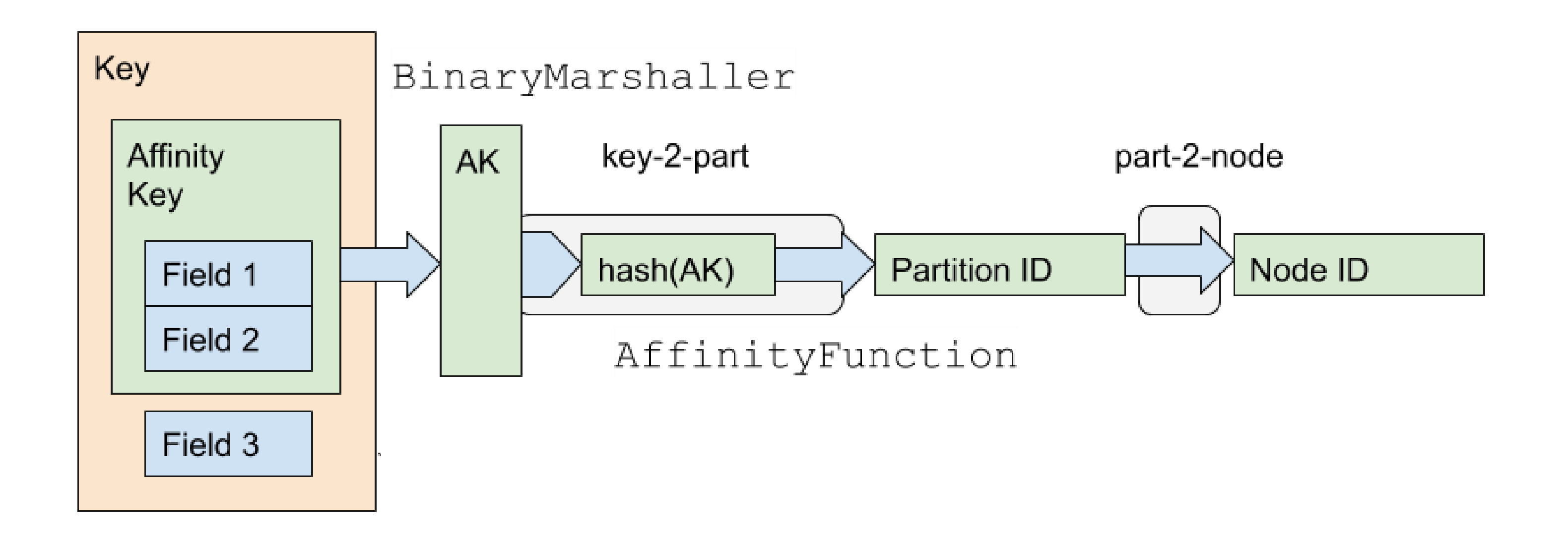
How to find some entry and update it

How the storage and redo log are organized

Over speed protection

Data location in the cluster
Affinity

Cache Key to node: Affinity Function



Affinity key
- is a key which will be used to determine a node



Cache Key to node: Affinity Function

Key

Affinity Key

Field 1

Field 2

Field 3

BinaryMarshaller

AK

key-2-part

hash(AK)

Partition ID

part-2-node

Node ID

AffinityFunction

Key value to be used in primary index (cache get)

Value

Field 1

Field 2

Field 3

Field value to be used in building index

Key (full) is needed for primary index inside partition.

Key and indexed fields serialized in the similar manner



Affinity function alternatives

- Naive : targetNode = K.hashCode() mod nodeCount
- Consistent Hashing (no more used in Ignite)

  https://en.wikipedia.org/wiki/Consistent_hashing

  http://theory.stanford.edu/~tim/s16/l/l1.pdf
- Rendezvous – used by Ignite

  https://en.wikipedia.org/wiki/Rendezvous_hashing

  http://www.eecs.umich.edu/techreports/cse/96/CSE-TR-316-96.pdf



Rendezvous or Highest Random Weight (HRW)

Minimizes rebalancing on nodes set changes (leave node, join node)

- Naive approach node = K.hashCode() % nodes
- Imagine node add event

Consider case 1 cache 1024 partitions

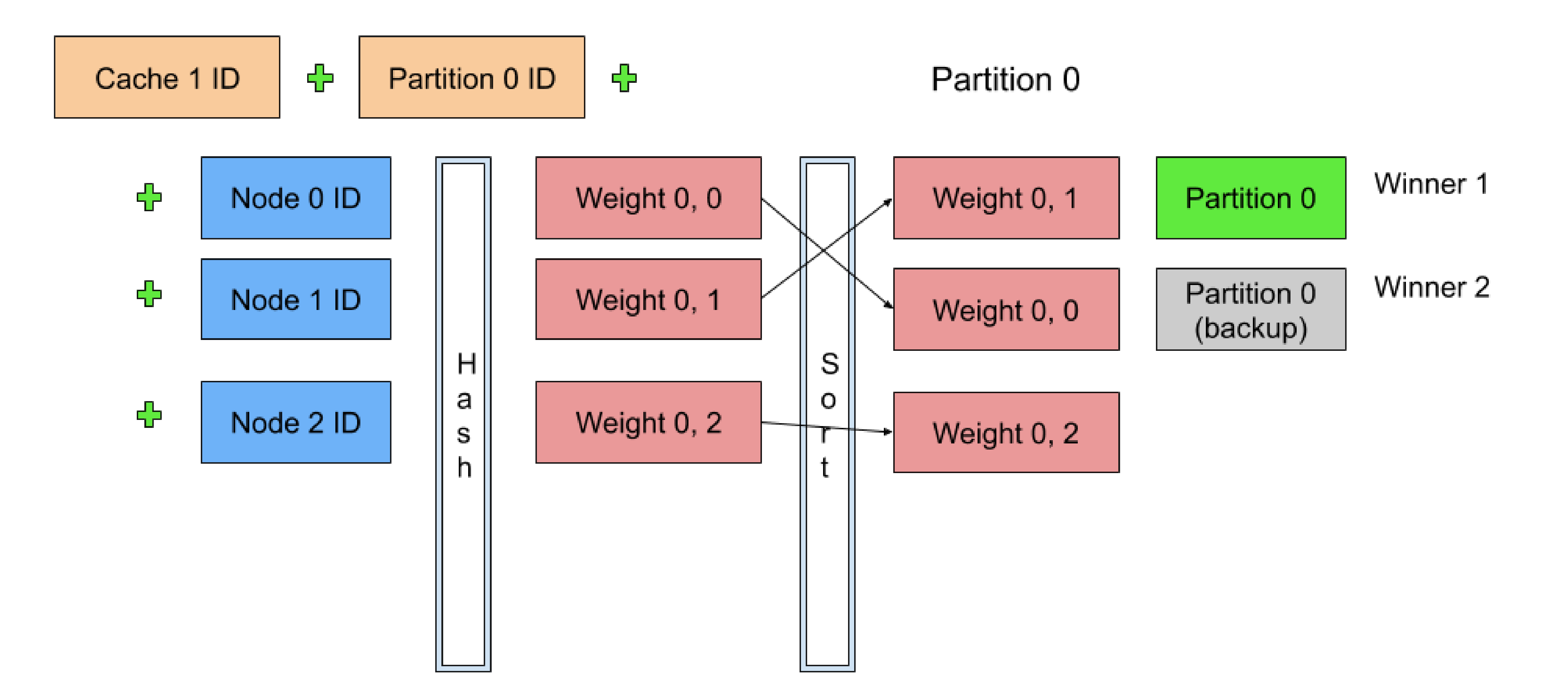
```
012345… %3 = 012012012…
012345… %4 = 012301230…
```

- Almost all partitions should migrate

Cache 0

Partition 0




Cache 1 ID
Partition 0 ID
Partition 0
Node 0 ID
Node 1 ID
Node 2 ID
Hash
Weight 0, 0
Weight 0, 1
Weight 0, 2
Sort
Weight 0, 1
Weight 0, 0
Weight 0, 2
Partition 0
Partition 0 (backup)
Winner 1
Winner 2

Cache 1 ID + Partition 0 ID +

Partition 0

+ Node 0 ID

+ Node 1 ID

+ Node 2 ID

Hash

Weight 0, 0

Weight 0, 1

Weight 0, 2

Sort

Weight 0, 1

Weight 0, 0

Weight 0, 2

Partition 0 — Winner 1

Partition 0 (backup) — Winner 2

Node 0

Partition 0 (backup)

Node 1

Partition 0

Node 2



Cache 1 ID + Partition 0 ID +

Partition 0

+ Node 0 ID
+ Node 1 ID
+ Node 2 ID
Node 3 ID

Hash

Weight 0, 0
Weight 0, 1
Weight 0, 2
Weight 0, 3

Sort

Weight 0, 1
Weight 0, 0
Weight 0, 2
Weight 0, 3

Partition 0 — Winner 1
Partition 0 (backup) — Winner 2
? depends on new winner

Node 0
Partition 0 (backup)

Node 1
Partition 0

Node 2

Node 3



Affinity gives ideal topology.

(Cache partition target node.
Actual data may be on the way)

- Rebalancing: Moving data from one node to another
- Actual get should go to old node until rebalancing finished
- GridDhtPartitionFullMap (simply - node2part)
- Indexes affected during rebalancing

Node 0

Node 1

Cache C (Partitioned)

C1 C3 C6

C2 C4 C7

Index C, field 1

Index C, field 2

Index C, field 1

Index C, field 2

Node 2

C3 C7

Index C, field 1

Index C, field 2



AGENDA

What is Apache Ignite and caches

How to find an ideal match node

How to find some entry and update it 

How the storage and redo log are organized

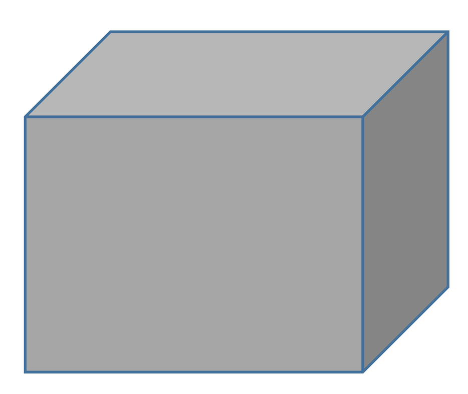
Over speed protection

LOCAL node:
Locating a key

Let's move closer to the disk

- One node
- One cache
- One partition
- One key
- We have some value to find
- Key is serialized by a marshaller (usually – binary)
- Split value (and key) - to chunks/blocks/pages





Cache 0

Partition 0

Key 0 | Value 0



All HDDs are block devices

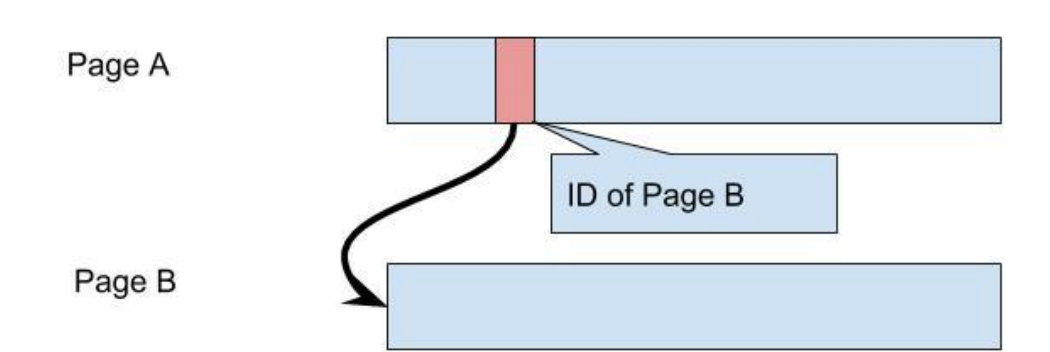
Durable (Page) memory

Pages Identification

- 4k
- Index – int: 0,1,..
- (+) Partition ID = Page Id

- Links ~ “Pointers”
- Links survives memory-HDD-memory

  (not depend on real address)





How to find a key in our pages

- B+ Tree
- Read optimized
- ~ Linked list (from pages) with top levels
- Order of iteration - preferable for range lookups

- PK/Primary Index
  - for each partition
  - Key Hash based
  - Key value compared (collision resistant)
- Secondary index: value or value start in Index page.



B+ Tree

Root Page

K4

Page Id

Keys < K4

Keys >= K4

Child Page

K1

K2

K3

K4

K7

K8

K9

Keys < K1

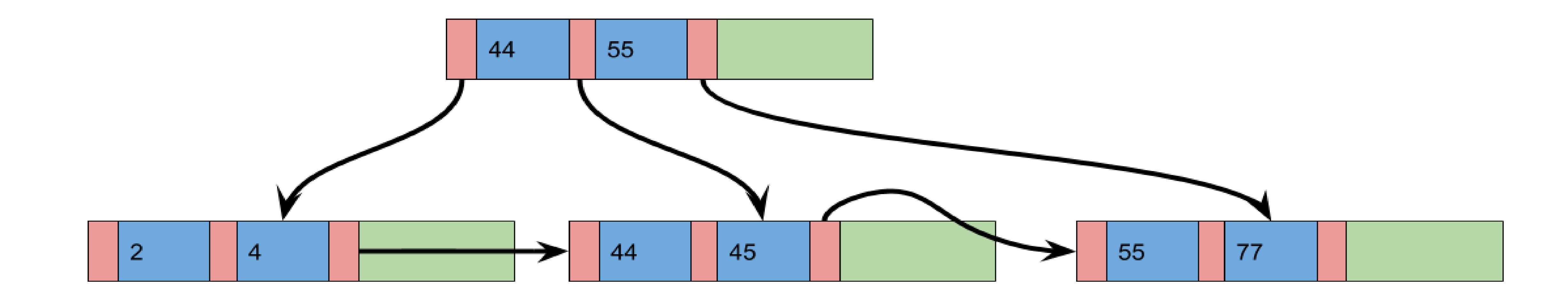
K1 <= ... < K2

K2 <= ... < K3

Value



B+ Tree



https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html



LOCAL node:
Store Values

Pages are allocated within region randomly

Durable Memory

Memory Region 1

Memory Region 2

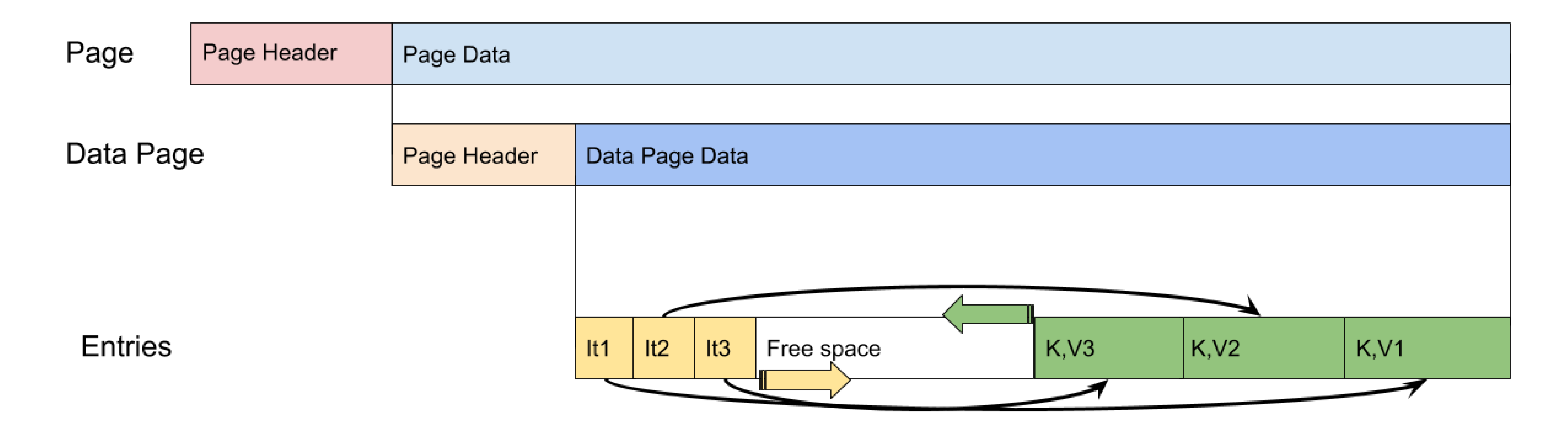
Memory Region 3

Memory Segment

Data page 1	B+ Tree Meta page 1	Index page 1	Index page 7	Data page 3	Index page 9	Free Lists Meta page 1	. . .	Page N

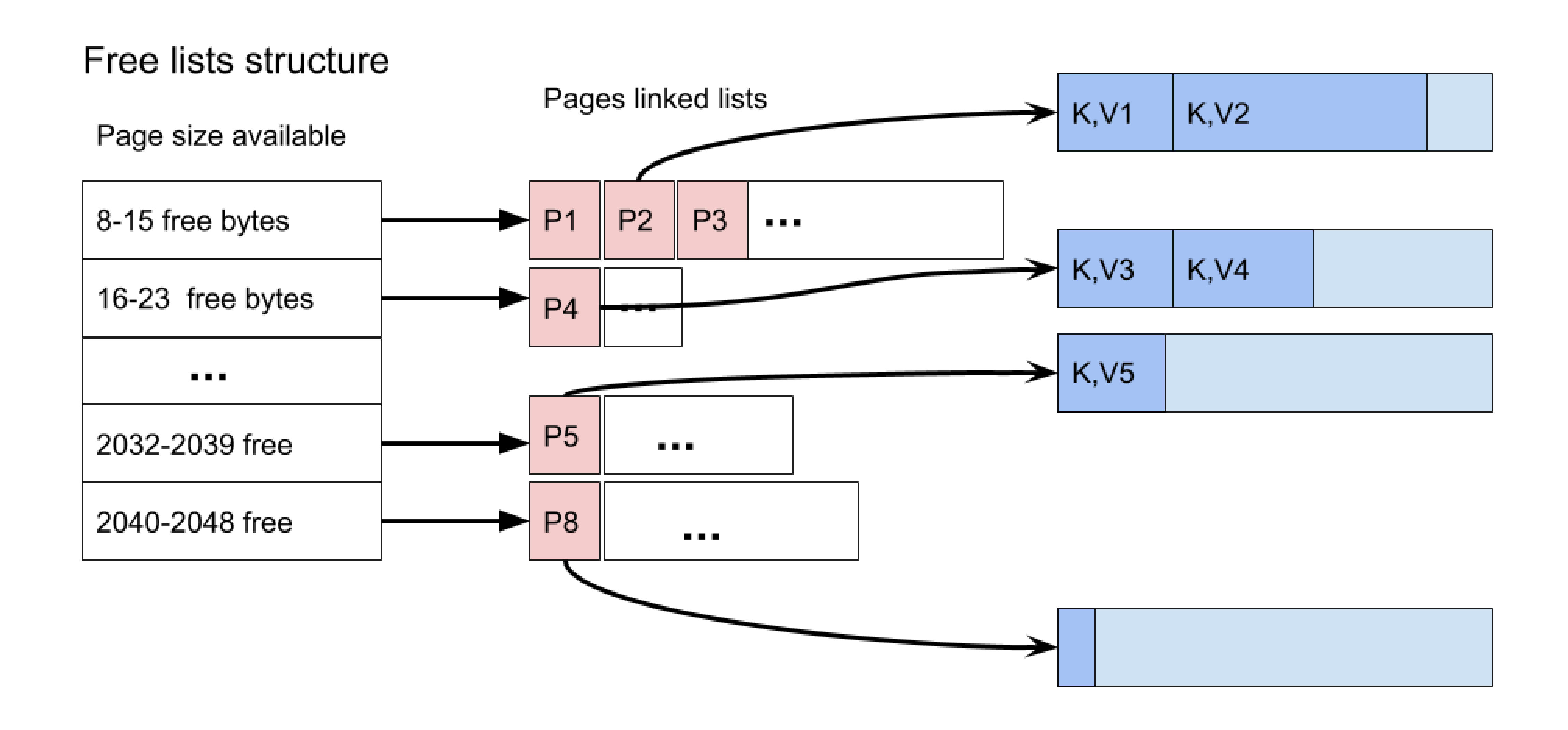


Data Page structure



K&V locator (in local node): FullPageId + item

Find suitable page for insertion data



Long objects

Free list

Page 4

Page 3 (allocated)

Page 2 (allocated)

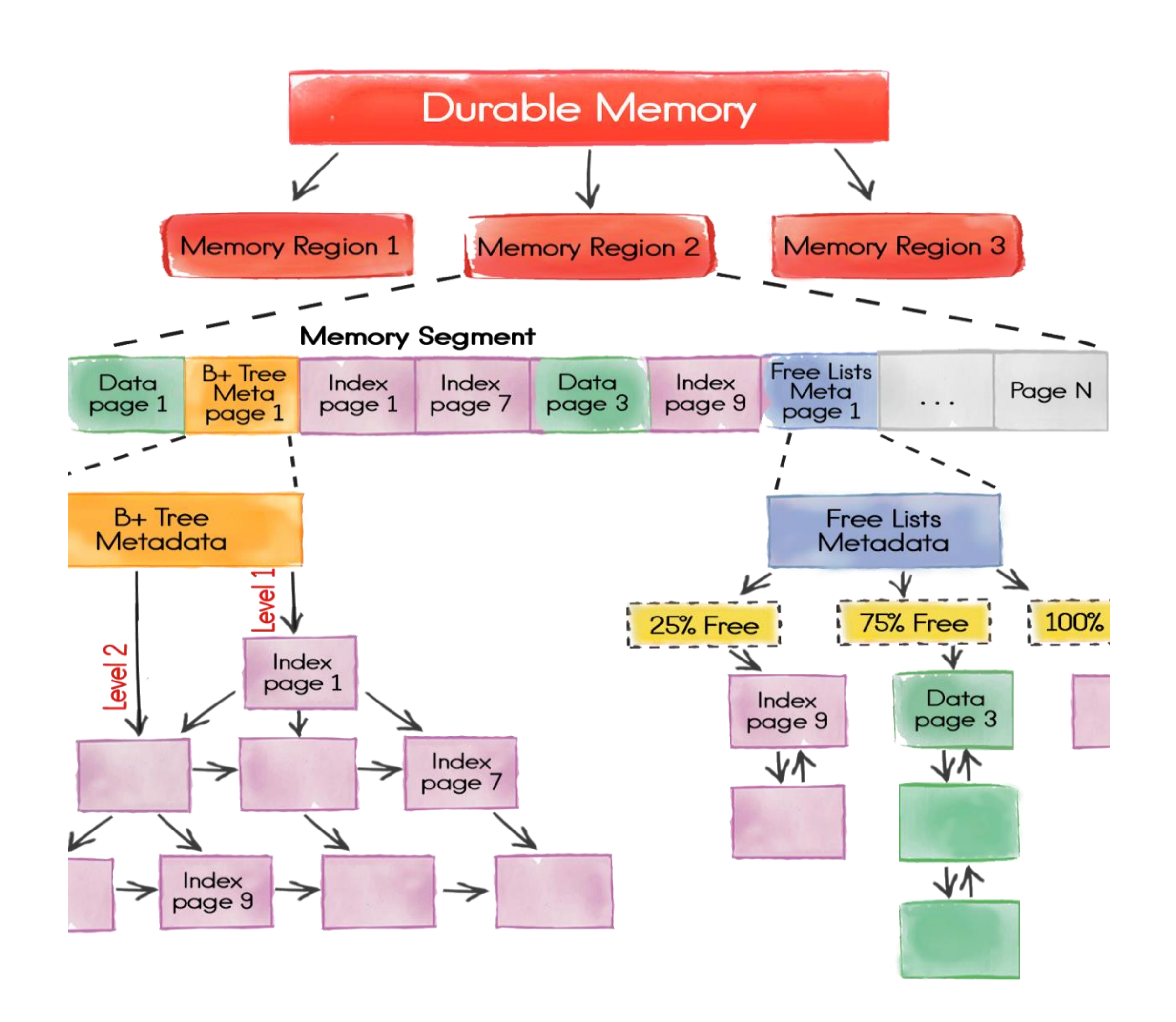
Page 1 (allocated)

Object:



RAM structure overview

- Regions – as configured
- Region has segments (depending on CPU count by default)
- Segment = set of RAM pages
- Pages has types and different formats
- Pages are linked between each other (Cross segment)





Loaded Pages Map
Buckets

Segment Data

B1 B2 BN

Segment 1
P1 P7 P2
Page 1 Page 2 Page 7

Linear Probe

Segment 2
P3 P5
Page 3 Page 5

Segment N
P4 P6
Page 4 Page 6

DataStorageConfiguration
setConcurrencyLevel

IGNITE_LONG_LONG_HASH_MAP_LOAD_FACTOR
(capacity=loadFactor*maxPages in segment)



LOCAL node:
A Page Modify and Write

Page R/W operations in RAM

Page ID->real address in memory

-1 atomic operation: resolution of ID to address & lock page

Case of setting a field in the page:

- `AbstractDataPageIO#setFirstEntryOffset`
- `PageUtils#putShort`
- `GridUnsafe#putShort(long, short)`
- `sun.misc.Unsafe#putShort(long, short)`

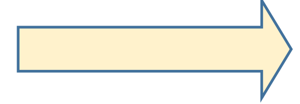


Read-Write (Memory & disc)

// Ignite classes: * FileIO

```
RandomAccessFileIO#write(ByteBuffer, long)

FileChannelImpl#write(ByteBuffer, long)
```



```
IOUtil#write(FileDescriptor, ByteBuffer, long,
             NativeDispatcher)

OUtil#writeFromNativeBuffer

NativeDispatcher#pwrite(…,
             (DirectBuffer)var1).address() + (long)var5
```



IO Util implementation

```
if (var1 instanceof DirectBuffer) {
    return
        writeFromNativeBuffer(var0,var1, var2, var4);
}  else {
    …
   ByteBuffer var8
        = Util.getTemporaryDirectBuffer(var7)
}
```



AGENDA

What is Apache Ignite and caches

How to find an ideal match node

How to find some entry and update it

➕ How the storage and redo log are organized ➕

Over speed protection

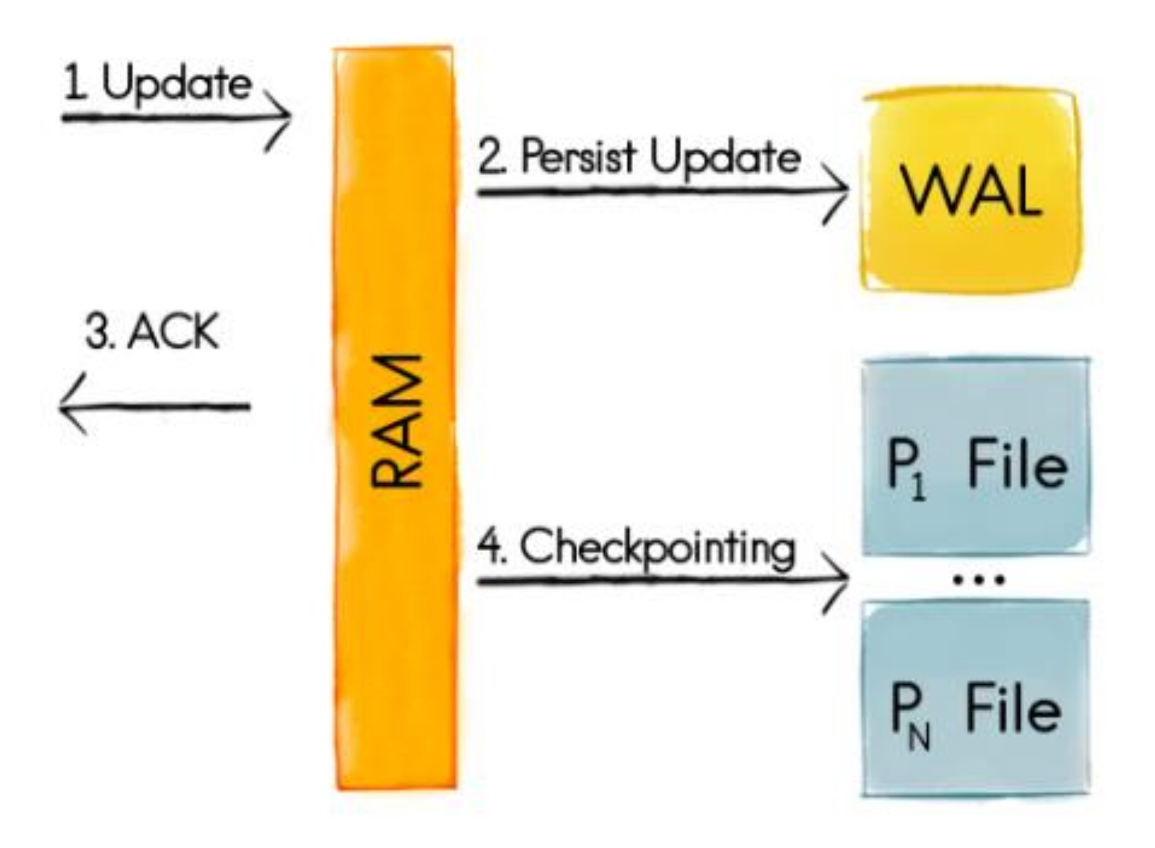
WAL

Write is not immediate after update

Data comes to
(1) In memory
Pages become dirty

(2) Write Ahead Log
(3) Update/TX completed

(4) Checkpointing = updating page store files = Background process





WAL

- WAL = **A**CI**D** – A&D, properties

https://en.wikipedia.org/wiki/Write-ahead_logging

- Both logical
  - Set user.lastSeen=…
- And physical
  - Change page PageID=…, at offset 4 to NNNNNNNN
-



Operation example

1. Set customer.lastSeen=xxx

K->AK->Partition->Node

BTree (primary idx)

Data Page

Data Page

Locking record & updating

2. Commit

WAL

...offloaded to HDD | Logical | Physical 1

FileChannel.force()
fsync()/fdatasync() in POSIX
*depends on mode



Why 2 types of records

- After checkpoint
- -1 CP history for data (logical)

- In the middle of checkpoint
- -2 CP for structure (physical)
- -1 CP for data (logical)

Persistent Store

part-0.bin, file

01

part-1.bin, file

Page13

Page14

part-2.bin, file

WAL Replay

WAL | Page snapshot | P. Delta | P. Delta | Page S.13 | Page S.14 | CP end | ...



More than 1 record for 1 put

- 1 logical, 1 physical
- Field changes its length + free lists updates
- Field update for 2+ pages for long objects
- Indexed field updated, need to add and remove index B+Tree nodes
- Updates in index may require Split-Merge operations for pages (2-3 page affected)
- Ignite tracks modifications, so Tracking page will be updated

- BUT: Latest Apache Ignite can share byte payload between records.



Sync strategies

- FSYNC – any case survive – OS crash and power
- LOG_ONLY – give the data to the OS - process crash
- BACKGROUND – by timer, some records may be lost

- Actual WAL is not one file
- Set of files = segments

- active, ready to be filled – Work
- Finalized – archive



Actual WAL structure

Fixed size
segment files

WAL Records log()

Worker threads:

T1. logs

T2. waits

Work
directory

00.wal

01.wal

02.wal

08.wal

09.wal

00...00

RR.RR

RR.RR

R

0

00...00

00...00

File Name:
N % WAL
Segment Size
(default 10)

Active file rotation

Archiver thread
Moves file from work

Archive
directory

45.wal

46.wal

47.wal

48.wal

RR.RR

RR.RR

RR.RR

RR.RR

RR

File Name: N.wal,
N - absolute segment index

WAL History Size,
Checkpoints

Deleted at
checkpoint end



Page store
(main storage for caches Data)

File Structure

- File per Partition
- Folder per cache
- WAL shared by all caches
- Indexes are shared by all partitions

Cache_A Files
P_1 P_{10}
P_{564} ... P_N
Indexes

...

Cache_B Files
P_1 P_{10}
P_{564} ... P_N
Indexes

Write-Ahead Log (WAL)

WAL Archive



Page store: File Structure

RAM

cache-C1

part-0.bin ..1.bin PN.bin index.bin

cache-C2

P0 P1 PN index.bin

WAL



Locate page placement: File Offset=Index*pageSize

Page Store

Cache “C1”

P0 P1 PN PI

idx=0

M M M M M

T T T T T

D

D

D

D

D

cache-C1

part-0.bin ..1.bin PN.bin index.bin

Page Store

Cache “C2”

P0 P1 PN PI

idx=0

M M M M

T T T T

cache-C2

part-0.bin ..1.bin PN.bin index.bin

WAL



Checkpointing

RAM to page store: checkpointing

Periodic action: timer or dirty pages percent

Fast STW collection of current dirty pages sets

It is our scope of data to be written to disk

Saved page: dirty = 0

Marking Pages
Cache “C”
P1
P2
PN
PI
M
T
D
Checkpoint pages: Set<Id>
Dirty pages: Set<Id>
checkpointLock.writeLock()



Process of writing

Page conflict during checkpoint => Copy on write.

CP buffer/pool

Overflow protected by exponential back off always enabled

Updated from worker

Checkpoint pages: Set<Id>

RAM

Page01 dirty *

Page13 dirty *

Checkpoint pool

Page 13

Copy on write
To CP pool

Copy of page to be checkpointed

Persistent Store

part-0.bin, file

Page 01

part-1.bin, file

Page13



AGENDA

What is Apache Ignite and caches

How to find an ideal match node

How to find some entry and update it

How the storage and redo log are organized

+ Over speed protection +

Average cp
write speed

Total pages to write

Pages left

Pages written and synced

Estimated cp
complete time

t

Actual progress

Total pages

Throttling zone

Overspeed

Estimated cp
complete time

t

Mark dirty progress



Files layout

Ignite work directory

binary_meta

{consistent id}

db

{consistent id}

cache-{cache-name}

cache-ignite-sys-cache

cp

wal

{consistent id}

archive

{consistent id}

marshaller

IgniteConfiguration
setWorkDirectory

Binary metadata
local file store

PersistentStoreConfiguration
persistenceStorePath

User
caches,
page store

PersistentStoreConfiguration
walStorePath

Checkpoint
markers

WAL work

PersistentStoreConfiguration
walArchivePath

WAL
archive

- Consistent Id –
  randomly or
  user specified node ID

Not covered

- Marshaller cache
- Binary Metadata



Summary

Do and Don't of today

- Don't write odd staff to your DB
- Length in bytes still matters
- Separate WAL & Page Store
- Don't set several nodes to share one HDD
- Use SSD where possible



(if you still want hardcore, join Apache Ignite community
dev@ignite.apache.org)

https://ignite.apache.org/community/contribute.html

Links

http://ignite.apache.org/

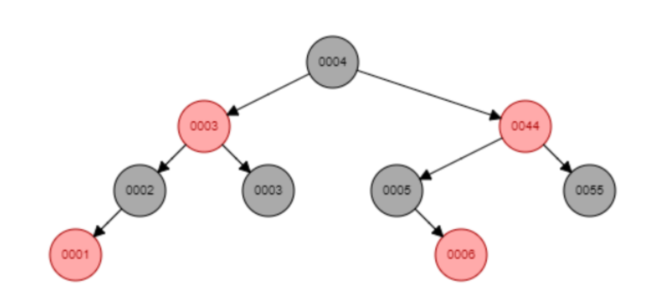
https://apacheignite.readme.io/docs/durable-memory

https://cwiki.apache.org/confluence/display/IGNITE/Ignite+Durable+Memory+-+under+the+hood

https://apacheignite.readme.io/docs/distributed-persistent-store

https://cwiki.apache.org/confluence/display/IGNITE/Ignite+Persistent+Store+-+under+the+hood

https://www.cs.usfca.edu/~galles/visualization/Algorithms.html





Should you have any question

dpavlov@apache.org

or just google

"Dmitriy Pavlov Apache"

Apache Ignite

How to use

user@ignite.apache.org

Related to contribution

dev@ignite.apache.org



May, 25, Online and free

